Week 3 Lab

Exercises will refer to my week 3 GitHub repo which is at: **https://github.com/oit-gaden/Web-Development-2019-Winter/tree/master/Examples/Week3/app2**

Don't forget to push your code to GitHub in a folder called week3 and your Docker image to Docker Hub.

Exercise 1 - Install Node and NPM

1. Install Node from https://nodejs.org/ (note: NPM is installed along with the installation of Node)
2. Open a command/shell window and run "**node --version**" to make sure you have successfully installed Node.
3. In the same command/shell window run "**npm --version** to make sure you have successfully installed npm.
4. Run "**npm help**" to see the several commands available. You can also find documentation at https://docs.npmjs.com/cli-documentation/

Exercise 2 - Create Week 3 Version of Your Web App

1. Create a Week 3 folder (where your Week 1 and Week 2 folder are).
2. Copy the entire contents of your Week 2 folder to your Week 3 folder. Just include the contents of your Web app.
   Don't include anything from the extra credit (Flask app) if you did that.
3. Copy the following from my week3/app2 folder in my GitHub repo to your Week 3 folder:

   .dockerignore
   .gitignore
   gulpfile.js
4. Change to your Week 3 folder and run "**npm init**". You can take all of the default answers or fill in answers as you wish.
   This will create the package.json file in your Week 3 folder.

Exercise 3 - Install Build Tools for Your Web App

1. Add the gulp CLI (command line interface). Open up a command/shell window and run "**npm install -g gulp-cli**".
   This is not strictly just for your current project but allows you to use the gulp command for all projects. To check to make
   sure gulp CLI is installed run **gulp --version**.
2. Install gulp and gulp plugins used for development build. You'll want to install all of the "devDependencies" in my "package.json"
   file. To do this, open a command/shell window in your week3 folder and run "**npm install --save-dev [package-name]**". For example,
   to install dependency "gulp", run "**npm install --save-dev gulp**". Check your package.json file to make sure all of the "devDependencies"
   listed match what is my package.json file.

Exercise 4 - Build Your Web App

1. Invoke VSCode (or your editor of choice) for editing files in your Week3 folder. You should be able to just open a command/shell window in your Week3 folder and type "**code .**".

2. At the top of each .html file surround your .css link line with the same removeIf and endRemoveIf HTML comment lines that are at the top of my example's index.html file. This causes your development time only .css links to be removed. The next step will replace them with the build version.

3. At the top of each .html file add the same inject and endinject HTML comment lines that are at the top of my example's index.html file. This causes the build version of your .css to be "injected" into your .html files.

4. Edit the gulpfile.js file. You'll see number of folder references to where my app files are located under "app/ ...". You'll need to change the folder references to point to where your files (html, css, images) are located. For example, if your html files are in the root of your week 3 folder then your path would be just "./". If you have .css files in a subfolder .css then the CSS folder would be "./css/".

5. Add jQuery as a runtime dependency. Run the command "**npm install jquery**". Check your package.json file to see if it was added under "dependencies".

6. Run the command "**gulp**" in your Week3 folder to build your app. You shouldn't see any errors displayed. You should see "Starting 'watch_files'". You can hit "Control-C" to exit from the gulp command watching files.

7. Using your open VSCode or the file explorer, look for the folder "**build**" and open it up. You should see your images copied to the images folder, the file app.min.css in the css folder (this is the bundled, minified version of your .css files) and the file vendor.min.js in the js folder (this is the bundled, minified version of your JavaScript runtime dependencies of which we only have jQuery. I would encourage you to look at the app.min.css file and see the minification applied (basically removal of whitespace).

8. View the gulpfile.js file and see if you can figure out how these files were copied/created.

9. Using VSCode, you should now be able to click on your home page in the build folder, run live server and see your home page displayed. You should also be able to navigate around your Web app using the page links at the top of each page. Styling from your .css file should be applied as expected. Stop live server.

Exercise 5 - Add Logic to Your Web App

1. Create a folder in your week 3 folder to contain your JavaScript and name it js.

2. Create a JavaScript file called students.js in the js folder.

3. Create functions in students.js like what you find in my products.js. The products.js basically reads a JSON string and uses it to populate the students table, using jQuery, rather than hardcoding the data in the table. You'll need to adapt the handling of products to the handling of students in your application. You'll also need to create a JSON string for the list of students. You might want to refer to https://www.w3schools.com/js/js_json_intro.asp to become familiar with JSON syntax. You might also want to refer to https://www.w3schools.com/jquery/default.asp to help with understanding the jQuery being used.

4. Edit your students.html file and add the code you'll find in my products.html lines 36-47 just before the body end tag.
   You'll need to change the names accordingly.

```
35
36          <!-- inject:js -->
37          <!-- endinject -->
38
39          <!--removeIf(production)-->
40          <script src='js/products.js'></script>
41          <!--endRemoveIf(production)-->
42
43          <script type="text/javascript">
44              $(document).ready(function(){
45                  initializeProducts();
46              });
47          </script>
48      </body>
```

5. Create a JavaScript file called login.js in the js folder.

6. Create functions in login.js like what you find in my registration.js. The login.js basically checks to make sure username and password
   have been entered and displaying an error if not. You'll need to adapt the handling of registration to the handling
   of login in your application.

7. Edit your login.html file and add the code you'll find in my registration.html lines 36-48 just before the body end tag.
   You'll need to change the names accordingly.

```
36
37          <!--removeIf(production)-->
38          <script src='js/registration.js'></script>
39          <!--endRemoveIf(production)-->
40
41          <!-- inject:js -->
42          <!-- endinject -->
43
44          <script>
45              $(document).ready(function() {
46                  initializeRegistration();
47              });
48          </script>
49      </body>
```

8. Rebuild your application.

9. Look at the file "**app.min.js**" created in the build/js folder to see the bundling (concatenation) of your
   two JavaScript files and the minification applied.

10. Using VSCode, you should now be able to click on your home page in the build folder, run live server and see your home page displayed. You should also be able to navigate around your Web app using the page links at the top of each page and see the . Styling from your .css file should be applied as expected. You should be able to see the students table dynamically populated from the JSON and the login should check the presence of username and password entries. Stop live server.

Exercise 6 - Build a Docker Image with Your App Build

1. Edit the Dockerfile and to the following:

   Replace the line: "**COPY . /usr/share/nginx/app/**" with "**COPY ./build /usr/share/nginx/app/**"

2. Build the Docker image.

3. Run the Docker image to test.

4. Push your Docker image to Docker Hub.

Extra Credit

Find another NPM package at https://www.npmjs.com/ and use it in your application to add functionality.

Other Things to Try (No extra credit)

1. Completely remove the node_modules folder and then run "**npm install**" in you Week 3 folder. See that the node_modules folder gets recreated as expected.

2. Run gulp again and repeat Exercise 4, step #7. Try editing page content and/or CSS styling and see the changes automatically refreshed in the browser.

3. Add the "scripts" from my package.json file and try them by running "npm run [script name]".

4. Looking at the package.json file you'll see a "^" prior to each version number. You can find out what that means at http://www.semver.org.